# 2. Subsystem Monitoring / Trouble Shooting Techniques and Utilities

## *2.1 COTS Management*

- **How will the operator know when a COTS process has gone down?**

If Sybase has gone down, there will definitely be error messages about Sybase connections in the logs; however, there can be error messages about Sybase connections for reasons other than the database being down. The operator should check the status of the Sybase SQL Server processes via the command line.

If an Autosys processor has gone, the only way that the operator can determine that is to run the command line check:

chk_autosys_up

## *2.2 Request Management*

- **For each subsystem, how can operators determine if a server has become quiescent (i.e., has finished processing all requests)?**

**Ingest -** If the logs are not being updated.

**PLS -** You can tell if the SubMgr is idling simply by inspecting the logs. The SubMgr checks its TIMERS on a set interval. If that is all that is seen on the last line of the log, then the SubMgr is idling.

**IDG -** For SRF-based servers such as subscription server and DAR communication gateway, quiescent means only SRF messages are shown in the log file. For LandSat7 gateway, EmailParser gateway and MOJO gateway, quiescent mean a repeated message pattern is shown in the log file.

**Data Distribution -** This is usually detectable because the log (debug, ALOG, or both) stops.

- **How can the operator correlate requests across system and servers?**

**Ingest -** Within Ingest, there is a unique RequestID for each request. Each request gets split into granules, each with a different GranuleID. When an rpc is sent to Stmgt or Sdsrv, the rpc ID contains the Ingest RequestID and

GranuleID as follows: IngestRQxxxGRyyy where xxx is the RequestID and yyy is the Granule ID.

**IDM -** IDM connections to servers can be determined via log output showing client paths and attempts at connections. Thus the server in question and the state of that connections can be determined, in general from either log at highest debug level at least.

**Storage Management -** Within STMGT, the best technique is tracking the RPC ID through the Debug logs. Note, however, that there is no practical technique for tracing requests through STMGT in the absence of debug levels set at 3.

**Data Distribution -** If things go right, simply by looking at the GUIs. Otherwise, the process is to look at the debug logs.

**IDG -** Within EcSbSubServer, there is a unique RPC ID, UUID+SBSRV, for each request to SDSRV and a unique RPC ID, UUID+SDSRV, for each request from SDSRV.

For LandSat7 Gateway, EmailParser Gateway and MOJO Gateway, the message protocols specify each request. Between DAR Comm Gateway and MOJO Gateway, the name of DAR API can track requests.

- **How can the operator determine which requests are hanging?**

**Ingest -** By seeing what the RequestState and DataGranuleState are in the Ingest database – can use the Ingest Monitor and Control GUI window to view these. If the RequestState is New, then that means that the request has not been received by Request Manager. If the RequestState is Active, then the request has been received by Request Manager. If the DataGranuleState is New, then the granule either has not been received by Granule Server or is in the transfer state. If the DataGranuleState is Transferred, then the granule is being preprocessed or is waiting to be preprocessed (only one granule can be preprocessed at a time per Granule Server). If the DataGranuleState is Preprocessed, then the granule is being archived by SDSRV.

Some granules get preprocessed and archived twice (i.e. Landsat-7, GDAS_0ZF, etc.). So if the DataGranuleState is Archived, then either the granule is done being processed or it is being preprocessed for the 2$^{nd}$ time or is waiting to be preprocessed for the 2$^{nd}$ time. Just because a granule stays in the same state for a long time, it does not mean that the granule is hung. It could just mean that the archiving or ftping is taking a long time.

**PLS -** If the SubMgr does not print out the results of the inspect from SDSRV after getting a notification for a datatype which PLS has a subscription for then

the SubMgr might be hung. It is possible to determine which granule the SubMgr is hanging on by looking at the PlNotification table which should contain the granule UR. From there you can determine which granule should match the granule from SDSRV by comparing its attributes as listed by SDSRV to the granules in the PDPS database under the PlDataGranule table. Once you have determined which granule best matches the one coming from SDSRV you can take the granuleId and perform a select in PlDprData to determine which DPRs are waiting for this data.

**Science Data Server -** Operators can determine if requests are hanging if they never see the rpc returned message in the debug log. If an acquire request is suspended with errors in the Data Distribution Server, the request will be hung until an operator intervenes. If the request is cancelled, then SDSRV will fail the request.

**Data Distribution -** If the requests have been instantiated by DDIST, they should show up on the GUI; else, SDSRV has to track down missing requests it sent.

### *2.3 Log Management/Usage*

- **What information is provided at each debug level, and which level should be used in various circumstances? How much performance degradation is expected at each level?**

  **Ingest -** Most debug information is printed at debug level 3.

  **Science Data Server -** The Debug level for SDSRV and the HDF EOS Server should be set to 2. Level 2 will display messages when SDSRV is making a RPC to another server or SYBASE. When the Debug level is set to three, a lot of metadata is output to the log file. For searches of granules that have a big descriptors (Landsat), the difference in the search time can be as much as 30 times more if the debug level is set to 3.

  **IDM -** Currently under review with regard to implementing ‘debug level’ appropriately. At the moment the highest level is the only useable mode. Luckily for IDM this is not a performance issue.

  **Storage Management –** The debug information output from Archive Server and Staging Monitor server falls in the following general categories :

  level 0 - all debug is turned off

  level 1 - all errors are recorded in the debug log

level 2 - all major events such as Archive create, Archive store, Archive retrieve are recorded. [Note Staging Monitor has no specific level 2 recording]

level 3 - this is a full trace recording of everything that is happening.

Debug level 3 is a large amount of output that might affect performance. Level 3 must be used when trying to document an NCR. Level 2 should be run if you want a general idea of what the server is doing at a given time. Level 1 should probably always be on when not running in production so that errors can be seen.

For Pull Monitor:

In general errors should only be sought after if a problem arises. Errors should be taken into account depending on the nature of the problem, the time it happened and other relevant clues. The messages are usually descriptive.

EcDsStPullMonitorServerDebug.log output:
level 0 - nothing useful for debugging (better to look in the ALOG)
level 1 - PF_VERBOSE statements (24 calls)
level 2 - PF_VERBOSE + PF_STATUS statements (26 calls)
level 3 - PF_VERBOSE + PF_STATUS + PF_DEBUG (526 calls)

PF_VERBOSE statements just give the main operational status of a procedure along with error messages.
PF_STATUS gives very little more in PullMonitor
PF_DEBUG gives a lot of data on operation, success or failure on a call and when the request enters and leaves a method.

PF_VERBOSE + PF_STATUS minimal usefulness for debugging

PF_DEBUG 13400:2000 (total no of lines of code in PMReal : approx no lines of code used by PF_DEBUG)

Most of the PF_XXX statements are cout or cerr statements. I don't know how this compares to the overall performance/load of the server.Pull Monitor, because it is single threaded, doesn't suffer much of a performance hit with debug output.

For all intensive purposes, debug levels for STMGT should only be set to 0 or 3.

In the ftp servers, almost everything is being logged as PF_DEBUG. The only messages Logged as PF_STATUS are ones dealing with server startup:

"DsStDCEServer constructor failed"
"DsStDCEServer start failed"
"Unknown exception during startup"

There are no messages being logged as PF_VERBOSE, although, during transfer the ftpclient library prints out to screen messages dealing with creating a ftp connection and writing blocks over to the destination. These messages get printed out to standard output therefore they get printed out all the time (equivalent to PF_VERBOSE).

**Data Distribution -** Like Storage Management, either run with Debug Level set to 0 for producing no debug information or run with Debug Level set to 3 for all debugging output.

**IDG -** Level one information is for information. Level two information is about sequential transactions during a normal processing thread. Level three information is about debug information and may contain more detailed information. How much performance degradation is expected at each level? Set DebugLevel to 3 will increase the startup time for subscription GUI to load subscriptions and events from database.

- **How can you tell when another subsystem is down using the debug logs?**

**Ingest -** When there are repeated rebinding messages.

**Science Data Server -** You can tell when another server SDSRV is attemptint to talk to is down by determining if the rpc is returned. If you set the Debug level to 2, SDSRV will log when a rpc is sent to a server. If the rpc is successfully received, you will see the rpc returned message. If you do not see the message, then it is a good chance that the other server is down.

You may see "DCE Rebinding Area Error: No more bindings (dce / rpc)" if SDSRV is unable to talk to another server. You will usually see the DsDdRequestMgrCFhExecutor message in the debug log if SDSRV can not talk to DDIST.

**IDM -** Check for retries of remote procedure calls or even failures in the debug log. In the event of failure the error back to the client (see above) is the starting point.

**Storage Management -** When any client of STMGT crashes, the server to which the client was attached will report a rundown invocation. The rundown

can typically be traced back to an RPC ID, which will reflect the client ID in the subpart of the RPC ID. (SDSV is SDSRV, DSDD is DDIST, IN* is Ingest.) If a STMGT server has crashed, any STMGT clients of that server will attempt to rebind, reporting a series of rebinding attempt messages.

**Data Distribution -** Usually because the last entry (or almost the last entry) in the log is a stmgt call; i.e., it has the DsSt prefix.

**IDG -** When there are repeated rebinding messages.

- **Under what conditions, if at all, should ECS Assist be used to monitor the status of logs or servers? What are the alternatives?**

Under no circumstances should ECS Assist be used to tail the logs – each log window consumes system memory in excess of the entire log size! Use xterms running tail –f commands in lieu of ECS Assist for log monitoring.

- **What's the recovery procedure when the ALOG reaches is maximum size?**

In this case, the server truncates everything in the log up through the time that it reaches the maximum (in effect, it "rewinds the file") and restarts the log.

- **What should the operator look for in the syslog as an indicator that things aren't working correctly (e.g., automount failures**)?

Out of swap space can cause servers to core dump. When starting servers, if no information is written to the logs, then the syslog may show what the problem. For example, missing a config file parameter needed by PF.

- **What are "normal" errors vs. errors the operator needs to take action on?**

**Science Data Server –** Below are the normal messages you should see in the Debug log on an acquire:

DsGeESDT::CreateSizedStagingDisk – Sending rpc to Staging Disk
DsGeESDT::CreateSizedStagingDisk – Staging Disk rpc returned
EcUtStatus.Ok() = 1 (if the status is 0, we did not successfully talk to the Staging Disk Server)
DsGeESDT::StagingDiskSaveOnExit – Sending rpc to Staging Disk

DsGeESDT::StagingDiskSaveOnExit – Staging Disk rpc returned
status = 1 ( if the status is 0, we did not successfully talk to the Staging Disk Server)

If you do not see the rpc returned message, then the request is hung trying to talk to the Staging Disk server. You should check the Staging Disk Server logs to see if the request was received. You can search using the RPC id.

Once the SDSRV talks to the Staging Disk Server, it talks to the Distribution Server. You should see the following messages in the Debug log:

DsSrWorkingCollection – CreatingDsDdRequestMgrC to DDIST
DsSrWorkingCollection – DDIST DsDdRequestMgrC returned
DsSrWorkingCollection – Sending rpc to DDIST
DsSrWorkingCollection – DDIST rpc returned
EcUtStatus.Ok() = 1 (if status is 0 then the Distribution Submit failed).

If you never see the rpc returned message, then SDSRV did not successfully talk to the Distribution server. You should check the DDIST logs and Gui. If a request is suspended with errors, the DDIST rpc never returns. The request must be resumed and cancelled to free the thread in SDSRV.

Below are the normal messages on an Insert

DsGeESDT::ArchiveStore – Sending rpc to Archive
DsGeESDT::ArchiveStore – Archive rpc returned
EcUtStatus.Ok() = 1 ( if the status is 0, we did not successfully talk to the Archive Server)

DsGeESDT::ArchiveDestroy – Sending rpc to Archive
DsGeESDT::ArchiveDestroy – Archive rpc returned

DsGeESDT::InsertMetadata – Sending metadata to the catalog
DsGeESDT::InsertMetadata – Catalog returned

If you never see the rpc returned message, then SDSRV did not successfully talk to the Archive Server. You should check the Archive logs.

**IDM -** All DMS errors will eventually be reported back to the client. The best place for the operator to intercept them would be the V0ToEcsGateway debug log. “Sent an error response back to the client” is the starting point for such troubleshooting.

**Storage Management -** The following are normal errors for Staging Monitor Server :

- on execution of stored procedure DsStSMRSelectByID the error message "Unable to get record with RequestID = " may appear because a new request is being processed.

- on execution of stored procedure DsStFLUpdAccessCount the error message "FileName [name} with CacheId [number] does not exist" may appear when trying to make an early update to a file access count to keep it in read only cache.

- on execution of method DoAdjustAccessCount the error message "*** Failed to update access count for file [name] may appear when trying to make an early update to a file access count to keep it in read only cache.

The following are normal errors for Archive Server :

- during Startup if the Staging Monitor is not up the following normal message appears "**** Failure connection to Staging Monitor during startup"

- on exection of stored procedure DsStGRDelete the error : "Error: cannot get parameter HWCI from the dictionary " will occur.

When any client of STMGT crashes, the server to which the client was attached will report a rundown invocation. The rundown can typically be traced back to an RPC ID, which will reflect the client ID in the subpart of the RPC ID. (SDSV is SDSRV, DSDD is DDIST, IN* is Ingest.) If a STMGT server has crashed, any STMGT clients of that server will attempt to rebind, reporting a series of rebinding attempt messages.

**Data Distribution -** The best approach to error messages is to use them when a request fails. Find the last “Failed…” message for a request, and read backwards through the log until you find some definitive information, usually a storage management error code and description.

**IDG -** A repeated login error or repeated user profile lookup failure means this user may not be a registered ECS user. He/she might be an intruder.

### *2.4 Statistics Used for Monitoring*

- **What database statistics are tracked by each subsystem? What are the SQL commands to get them? GUIs?**

**Ingest -** The Ingest GUI is used to access the database statistics. The Monitor and Control GUI window can be used to look at active requests and requests which have just finished. It displays the data provider, request state, granule states, data types, start and end time and size of the data. The History Log GUI window can be used to look at requests which completed. It can be used to display information about each request and its granules or statistical information. In addition to the information shown in the Monitor and Control GUI window, it also displays the time to transfer, preprocess and archive for each granule and request.

**Storage Management -** The STMGT GUI provides the capability for monitoring the cache statistics for the Pull Area and, for the next 4PY patch and beyond, for the Read-Only Cache. Operators can view the remaining space in each cache, as well as a list of files which are currently in the cache. For the Pull Area, the operator has the ability to force deletion of files from the pull cache. (This is not currently supported for the read-only cache from the STMGT GUI.)

### *2.5 Monitoring Granules*

- **Is there a technique or utility for matching browse data with granules?**

  We do not have a special script for matching browse data with granules. Below is the SQL to accomplish this.

```
select * from DsMdBrowse a
where a.dbID in ( select b.browseId from DsMdBrowseGranuleXref b,
DsMdGranules c
where b.granuleId = c.dbID)
```

### *2.6 Monitoring Subscriptions*

- **For SBSRV, how can the operator tell when notifications have been sent (or not sent)? Can logs show this?**

  There are two notification types:

  1. Normal Notification: Operator should find a debug message that notification for the subscription has been sent successfully by looking for information, such as, "Trigger: Succeeded" in the log file. But this message may still appear even an notification is not sent successfully. This is a known bug and will be fixed in 5B.

2. Message Passing Notification: This is basically the same as above. But since the notification is through message passing, the sender (SBSRV) doesn't know if the notification has been sent successfully or not. To check it, one way is to check from the subscriber's side to see if a notification has been received within a period of time.

### *2.7 Monitoring DCE*

- **How will the operator know when DCE has gone down?**

  DCE "going down" can involve either the DCE master server, one (or more) DCE client daemons on every host, or the DCE Cell Manager. There is no automatic notification when the client daemons have gone down. A symptom could be ECS Applications failing to communicate to other Applications.

  To check, login to the suspect host and run 'dceverify'. An output containing one of the following will be seen:

  "DCE Daemons: BAD – dced - not running!"
  "DCE Daemons: BAD – cdsadv - not running!"
  "DCE Daemons: BAD – dtsd - not running!"

  There is also no automatic notification when the DCE Servers have gone down. A symptom could be ECS Applications failing to communicate to other Applications. To check, login to any host that is not the DCE Master or Replica and run 'dceverify'. An output containing one of the following will be seen:

  " Security: WARNING: <*message*>"
  " CDS: Cannot contact CDS"

  The Cell Manager Tool requires a host agent daemon to be running on every DCE host it is monitoring. There is no automatic notification when these daemons are not running. To check:

  - From the Cell Manager tool, select the "Configuration Manager" button
  - From the Configuration Manager Window, select the suspect host and then the "DCE Status" button.
  - If the cell manager daemon is not running on the host, there will be a notification to "See Errors".
  - Selecting the "Show Errors…" button will display the errors and should include "The cellmgrd is not running."

### *2.6 Using HP OpenView*

- **Why is HP OpenView so slow?**

  The slow down is usually caused by poor performance on the host on which HP OpenView is running. Check the load by running the "uptime" command. If the load is over 2.5 or so, performance will take a hit. Run "top" to find out which processes are causing the problem and you may be able to shut down some non critical applications to free resources. (xnmevents processes can sometimes use enormous amounts of CPU when there are many events coming into HP OpenView, these can usually be shut down without impact.)

- **Why didn't a server start from HP OpenView?**

  First, if you still have the window up from which you started HP OpenView, watch the window for DCE related activity messages after you click the Start Executable drop down form an icon. If you don't see that, there is a good chance that there is a DCE/configuration problem on the HP OpenView server. Now let's assume that you did see DCE activity when you selected "Start Executable". Login to the server that the executable is expected to start on and 'cd /usr/ecs/CUSTOM/SHARED/logs'. (Make sure the server isn't already running first.) Then 'tail –f EcMsAgSubAgentDebug.log'. Try to restart the server from the HP OpenView GUI. If the DebugLevel is set to 3 in the subagent .CFG file, you should see a message showing that an attempt was made to start the custom code in the desired mode. If you see that, MSS has done it's part. The problem is likely in the startup script that is being called. Try running the script that you see listed in the Debug log manually on the command line. You should see any errors listed to stdout. If servers start from the program level, but not the application level, it's a safe bet that the application level startup script has an error in it.

- **Why didn't a server shutdown from HP OpenView?**

  Like the failure to start scenario above, if you have the window open from which HP OpenView was started, check for DCE activity when you select "Shutdown Executable". No output usually means a DCE outage or configuration error on the HP OpenView / Deputy machine. Login to the machine where the server is running and verify if it is still running or not. There could be a delay in HP OpenView updating the status of the icon representing the custom server. If the server is running, check the usr/ecs/CUSTOM/SHARED/logs/ EcMsAgSubAgentDebug.log and look for a message like: "MsAgDeputyGate::StopExecutable marking processes for shutdown" which will show that the SubAgent received the shutdown request. Now check the .ALOG for the server that you are trying to shutdown and look for a line that says something like: "Msg: [ServerName] shut down Priority: 2 Time : 03/03/99 18:32:30" and also looke in the xxxDebug.log file for a message that says: "EcAgManager::Shutdown called". It these messages

show up in the log, the server has received the shutdown request, but has not yet shut itself down. The server might be hung in its shutdown method.

- **Why am I missing wheat icons?**

  A likely cause is that the NFS mount that holds the active modes file is not properly mounted. The subagents read this file to determine which installed applications should be sent back to HP OpenView to be displayed as wheat icons. After fixing the NFS problem, a rediscovery is often necessary. See the next topic.

- **How do I force a rediscovery of installed applications?**

  Start the ModeManager GUI from the Misc heading of the menu bar and then click on the REDISCOVER button at the bottom left of the GUI. The rediscovery process can take several minutes.

- **How do I get rid of red icons?**

  Select “Remove Died Executables” from the Misc heading of the menu bar. Wait until you no longer see [Sync] displayed at the bottom of the HP OpenView Window. After [Sync] is no longer displayed, all red icons should be gone.

- **HP OpenView needs to be “reinitialized”.**

  See cleaning the object database (Recovery Tips and Techniques, above)